機械学習による分類

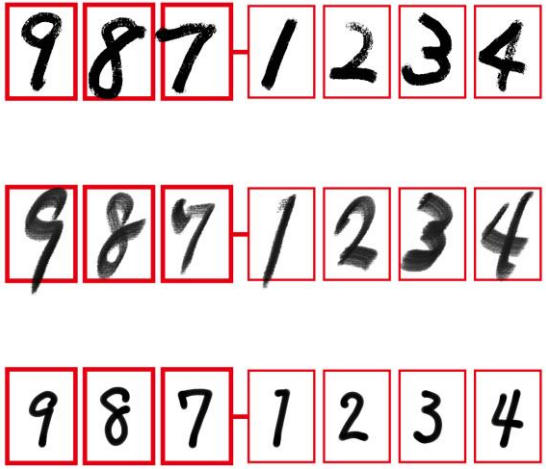
手書きの数字をコンピュータに認識させよう

手書きの字を自動でコンピュータに読み込みたい

数字を画像として読み込むだけでなく、数値として扱うことができれば便利になる。

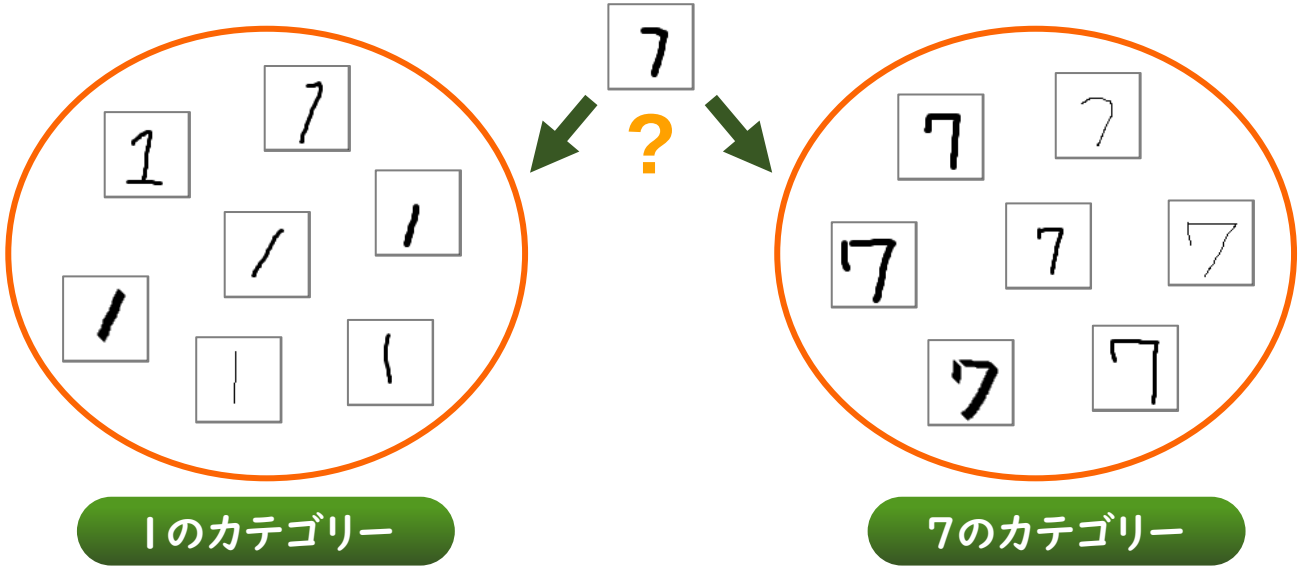
例）手書きの郵便番号

→住所がわかり配達先の仕分けができる



分類とは

データがどのカテゴリーに属するかを予測する。
そのために、事前に正解があるデータを使って学習しておく。
（教師あり学習）



分類を自動でできるようになると・・・

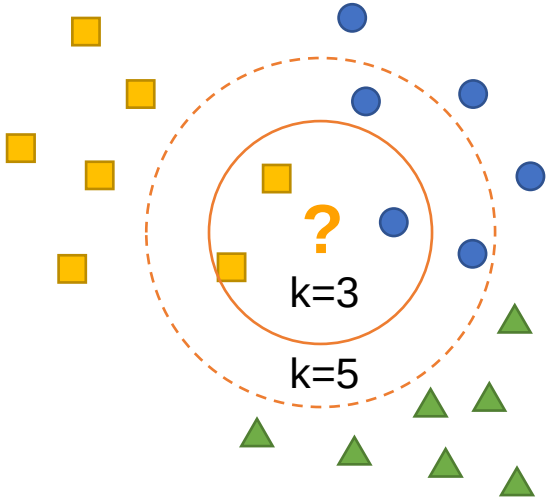
作業を自動化することができる

- 文字や音声の認識
- 自動運転での交通状況の確認
  （信号、他の自動車、歩行者、車線などを区別して認識）
- 病気の発見
  → 早期治療に結び付ける
- コンピュータウイルスの発見
  → 情報セキュリティの向上

分類のアルゴリズム①

k近傍法

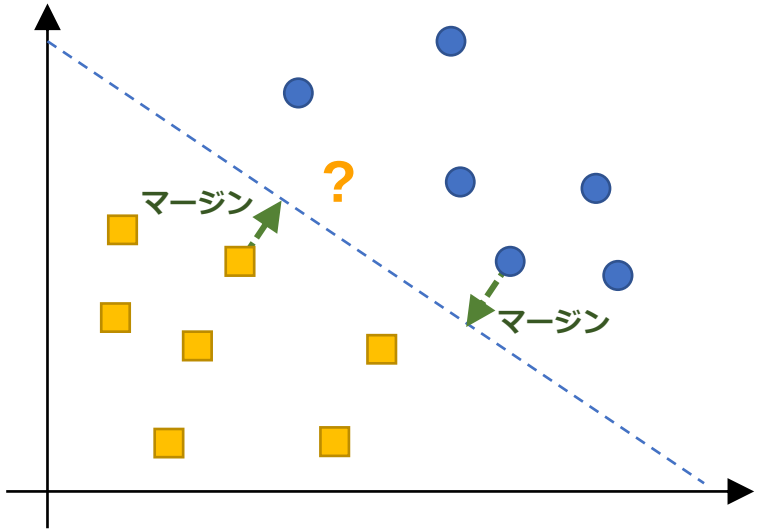
予測したいデータの近いk個のデータの中で、最も多いもののカテゴリーと推測する。



分類のアルゴリズム②

サポートベクターマシン（SVM）

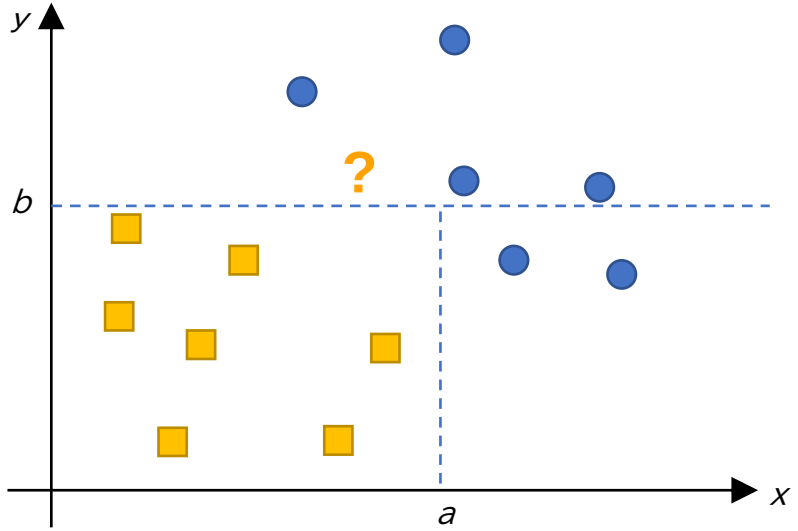
カテゴリを可能な限り分割する境界線を見つけて、分類する。
このとき、マージンが最大になるように境界線を引く。

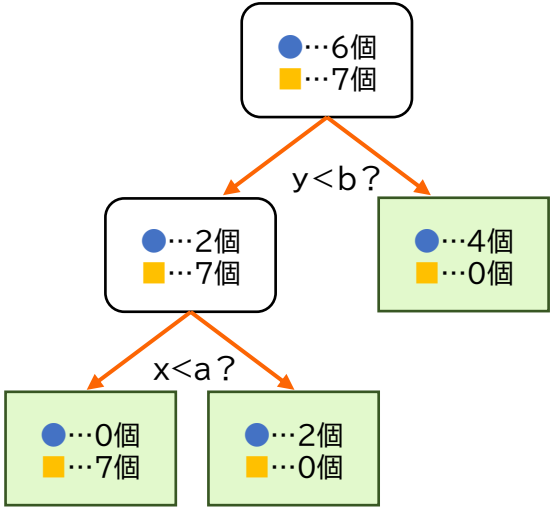


分類のアルゴリズム③

決定木

段階的にデータを分割してデータを分類する。
分析結果は、木のような形状で示すことができる。





分類の他のアルゴリズム

他にも

ナイーブベイズ分類器　　ロジスティック回帰

などのアルゴリズムがある。

学習データがどのような分布になっているかに合わせて、
アルゴリズムを選ぶ。

分類のモデルの評価

正解率（Accuracy）

正しく分類できた割合

$$\frac{\square\text{の中の数の合計}}{\square\text{の中の数の合計}}$$

		実際		
		A	B	C
予測	A	28	4	5
	B	0	26	3
	C	1	2	27

右の表では → $\dfrac{28+26+27}{28+4+5+0+26+3+1+2+27} \fallingdotseq 0.84$

他にも **再現率（Recall）** **適合率（Precision）** などがある。

手書き数字を分類してみよう

【分類の流れ】

1. 分類のモデルを作るためのデータを読み込む
2. データを用いて、学習モデルを作成する
3. テストデータを使って、モデルを検証する
4. モデルを使って、実際のデータを分類する

プログラムの解説

```
from tensorflow import keras
(X_train, y_train), (X_test, y_test) =
                        keras.datasets.mnist.load_data()
```

mnistという手書き数字のデータを読み込む

【訓練データ　60000字】
X_train（画素データ）
y_train（正解ラベル）

【テストデータ　10000字】
X_test（画素データ）
y_test（正解ラベル）

プログラムの解説

```
import matplotlib.pyplot as plt
```

matplotlibのpyplotというグラフを描画するモジュールを読み込む

```
plt.imshow(X_train[0], cmap='Greys')
plt.show()
```

X_train[0]の画素データをグレーの濃淡で表示する

```
print('正解ラベル', y_train[0])
```

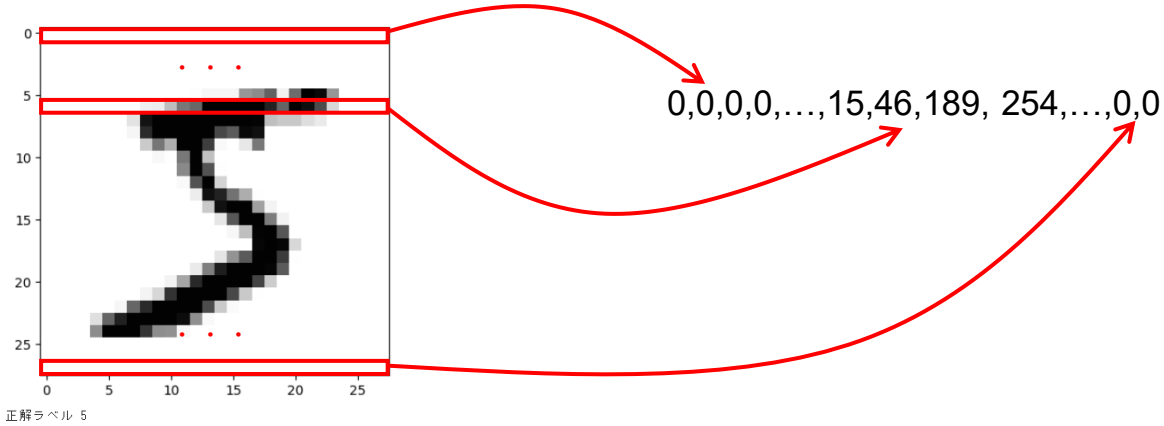
y_train[0]の正解ラベルを表示する

プログラムの解説

```
X_train = X_train.reshape(60000,784)
X_test = X_test.reshape(10000,784)
```

画素データを28×28の2次元配列から784要素の1次元配列に変換する

（X_trainは60000字、X_testは10000字）



プログラムの解説

```
from sklearn.neighbors import KNeighborsClassifier
```

scikit-learnに含まれる

k近傍法を行うためのモジュールをインポートする

```
model = KNeighborsClassifier(n_neighbors = 5)
model.fit(X_train, y_train)
```

k近傍法のk=5として学習を行う

プログラムの解説

```
from sklearn.metrics import accuracy_score
```

scikit-learnに含まれる

正解率を求めるためのモジュールをインポートする

```
y_model = model.predict(X_test)
```

学習したモデルを用いて、X_testを分類する

```
accuracy_score(y_test, y_model)
```

学習したモデルを用いた分類の正解率を求める

プログラムの解説

```
import cv2
```
画像処理のライブラリをインポートする

```
im = cv2.imread('pic.jpg')
```
「pic.jpg」というファイルを読み込む

```
im = cv2.resize(im,(28,28))
```
画素数が28×28になるように調整（縮小）する

```
im = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
```
BGRの3色をグレーに変換する

```
im = cv2.bitwise_not(im)
```
色を反転する

```
plt.imshow(im, cmap='Greys')
```
画像をグレーの濃淡で表示する

プログラムの解説

```
kekka = model.predict([im.reshape(784)])
```

画素データを784要素の1次元配列に変換して、分類する
引数は配列（リスト）として渡す

```
print('分類結果', kekka[0])
```

分類結果を表示する
リストとして渡したデータを分類するので、要素数1の配列として
結果が得られる

分類について、さらに学習を深めるには・・・

モデルについての理解を深める

kの値を調整して、正解率が高いモデルになるように調整する。

今回学習しなかったモデルについて学習を進める

ナイーブベイズ分類器、ロジスティック回帰など
分類の別のアルゴリズムを学習する。

分類を使った応用

こんなことに応用できます

- 迷惑メールかどうかを推測する。
- 新聞記事の文章をもとに政治記事か、社会記事か、スポーツ記事かを推測する。
- 小説の出現単語をもとに作者が誰かを推測する。